REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From – To)*
12 Jan 2015	Journal Article	May 2014 - October 2014

4. TITLE AND SUBTITLE

Learning SAS's Perl Regular Expression Matching the Easy Way: By Doing

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Paul Genovesi

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

USAF School of Aerospace Medicine
Aeromedical Research Department
2510 Fifth St.
Wright-Patterson AFB, OH 45433-7913

8. PERFORMING ORGANIZATION REPORT NUMBER

AFRL-SA-WP-JA-2014-0052

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSORING/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

Distribution A: Approved for public release; distribution is unlimited. Case Number: 88ABW-2014-4182, 4 Sep 2014

13. SUPPLEMENTARY NOTES

14. ABSTRACT

Perl Regular Expression (PRX) functions were added in SAS9. The key to learning how to use them is learning PRX metacharacter behavior within a PRX match. The more you practice PRX matching, the more proficient you become. Ideally, practice would involve a method for repeating the matching process so as to see cause and effect between the input, in the form of modifications to the perl regular expression and/or the source string, and the output in the form of the match results. But just as important as the practice itself would be a method for logging your "practice trail" in a file for future reference and expansion as well as avoidance of wheel-reinventing. But how do you do this without this file becoming bloated, cluttered, and unmanageable? The answer is to let it become bloated and cluttered.

Enter the regex_learning_tool consisting of a SAS Enterprise Guide project and an Excel file containing your practice trail in the form of match records each with a different perl regular expression and/or source string. The regex_learning_tool allows both beginner and expert to efficiently practice PRX matching by selecting and processing only the match records that the user is interested in based on a search of either PRX metacharacters contained in the perl regular expression or character string(s) contained in a match description field. The current Excel file contains over 400 match records demonstrating the use of most all PRX metacharacters. This paper will explain how to use the regex_learning_tool so that you can practice PRX matching, retain what you learn, and not have to worry about how bloated, cluttered, and unmanageable your practice trail becomes.

The regex_learning_tool project was written using SAS Enterprise Guide 5.1 and SAS 9.3 on a Windows 7 Enterprise operating system. A copy of the SAS Enterprise Guide project, Excel file, as well as other tool-related material can be found at sascommunity.org under Papers and Presentations.

15. SUBJECT TERMS

Perl regular expressions , prx functions , pattern matching , training , learning , practicing

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Paul Genovesi
U	U	U	SAR	13	19b. TELEPHONE NUMBER *(include area code)*

Paper RF-10-2014

Learning SAS's Perl Regular Expression Matching the Easy Way: By Doing

Paul Genovesi, Henry Jackson Foundation for the Advancement of Military Medicine, WPAFB, OH

ABSTRACT

Perl Regular Expression (PRX) functions were added in SAS9. The key to learning how to use them is learning PRX metacharacter behavior within a PRX match. The more you practice PRX matching, the more proficient you become. Ideally, practice would involve a method for repeating the matching process so as to see cause and effect between the input, in the form of modifications to the perl regular expression and/or the source string, and the output in the form of the match results. But just as important as the practice itself would be a method for logging your "practice trail" in a file for future reference and expansion as well as avoidance of wheel-reinventing. But how do you do this without this file becoming bloated, cluttered, and unmanageable? The answer is to let it become bloated and cluttered.

Enter the regex_learning_tool consisting of a SAS Enterprise Guide project and an Excel file containing your practice trail in the form of match records each with a different perl regular expression and/or source string. The regex_learning_tool allows both beginner and expert to efficiently practice PRX matching by selecting and processing only the match records that the user is interested in based on a search of either PRX metacharacters contained in the perl regular expression or character string(s) contained in a match description field. The current Excel file contains over 400 match records demonstrating the use of most all PRX metacharacters. This paper will explain how to use the regex_learning_tool so that you can practice PRX matching, retain what you learn, and not have to worry about how bloated, cluttered, and unmanageable your practice trail becomes.

The regex_learning_tool project was written using SAS Enterprise Guide 5.1 and SAS 9.3 on a Windows 7 Enterprise operating system. A copy of the SAS Enterprise Guide project, Excel file, as well as other tool-related material can be found at sascommunity.org under Papers and Presentations.

INTRODUCTION

The regex_learning_tool project imports match records contained in an Excel file and matches their regex field with their source field just as the prxmatch() and prxchange() functions would do within SAS. But the regex_learning_tool project is more than a simple matching tool. It allows the user to select which match records to process based on 2 different types of searches (a regex field search for metacharacters and a match_description field search for keywords). This feature is especially valuable for large, messy Excel files containing many match records.

This paper will discuss the following regex_learning_tool topics:

I. Setup and Running:
II. Design and Operation:
    1. How the project simulates SAS's prxmatch() function.
    2. How the project simulates SAS's prxchange() function.
    3. The input:
        - Excel file input_file.xlsx, which contains the match records and gets imported by the project.
    4. The output:
        - Both a SAS dataset and a SAS report contain the match results.
    5. The 3 process flows and 1 ordered list section, which make up the project.
    6. The autoexec process flow.
    7. The ordered list section.
    8. The relationship between ordered lists and SAS programs.
    9. How the project processes a substitution PRX match record.
    10. A regex field metacharacter search versus a match_description field keyword search.
    11. Why the project makes 2 passes through the imported data.
III. Effective Use Techniques:
    1. Commenting out a match record to avoid processing.
    2. Using the Modify-Save-Run-Repeat technique for quick learning.
    3. Creating a new match_description field keyword search.
    4. What to do if you get a field truncation error.

SETTING UP THE PROJECT

Instructions for setting up the regex_learning_tool project:

1. You need the following 2 items for setup:
    a. regex_learning_tool.egp

- The SAS Enterprise Guide project.

b. input_file.xlsx
   - The Excel file containing your match records.

2. Open SAS Enterprise Guide.
3. Open the regex_learning_tool project.
4. Click 'No' when asked whether to run the autoexec process flow.
5. In the Project Tree, click on the autoexec process flow.
6. In the autoexec process flow window, right-click on the input_file Excel icon.
7. Click on Properties.
8. Update the 'File name' field by clicking on the Change button and entering the path for where input_file.xlsx is located.
9. Now right-click on the autoexec process flow in the Project Tree and click 'Run autoexec.'
10. You are now ready to run the project.

RUNNING THE PROJECT

Instructions for running the regex_learning_tool project:

1. Open SAS Enterprise Guide.
2. Open the regex_learning_tool project.
3. Click 'Yes' when asked whether to run the autoexec process flow. Running this does an initial import of input_file.xlsx as well as compiling any macro programs and variables.
4. If the latest, saved contents of input_file.xlsx have been imported, then proceed to the next step and run a SAS program; otherwise, skip to step #7 and run an ordered list, which imports input_file.xlsx before it runs a SAS program.
5. In the Project Tree, go to one of the following process flows:
   - regex field Search-and-Match programs
   - match_description field Search-and-Match programs
6. Right-click on a program and click 'Run.' There are currently 44 regex field Search-and-Match programs and 13 match_description field Search-and-Match programs. Skip to step #9.
7. In the Project Tree, go to the ordered list section named:
   - Ordered Lists – Import, Search, and Match
8. Right-click on an ordered list and click 'Run.' There are currently 44 ordered lists that perform regex field searches and 13 that perform match_description field searches.
9. If you ran a "regex field search" ordered list or a "regex field search" program, then, at run conclusion, the project will:
   a. Move to the "regex field Search-and-Match Programs" process flow.
   b. Display the following 4 tabs for the program (i.e., the program with the same name as the ordered list that was run):
      1) Program
      2) Log
      3) Output Data
      4) Results
10. If you ran a "match_description field search" ordered list or a "match_description field search" program, then, at run conclusion, the project will:
    a. Move to the "match_description field Search-and-Match Programs" process flow.
    b. Display the same 4 tabs for the program as listed above in section 9b.
11. The program output (i.e., both the SAS dataset and SAS report) will be available for the remainder of your project session.
    a. Running only a program contained in either the "regex field Search-and-Match Programs" process flow or the "match_description field Search-and-Match Programs" process flow (as opposed to running the same-named ordered list) will not import data from the Excel file input_file and consequently, any changes made to input_file will not be processed. Every ordered list performs an import of input_file.
    b. Remember to save your latest changes to input_file.xlsx before running an ordered list in the regex_learning_tool project or they will not be processed.

HOW THE PROJECT SIMULATES SAS'S PRXMATCH() FUNCTION

There are 2 types of match records contained in input_file.xlsx:

1. PRX matches
2. Substitution-PRX matches.

If a match record in input_file.xlsx contains a regex field with syntax:

/<pattern>/

then

1. It is a PRX match.
2. The regex_learning_tool project will use a prxmatch() function to process it.

The following is from SAS Help and Documentation:
SAS's prxmatch() function:

1. Searches for a pattern and returns the position at which the pattern is found.
2. Has the following syntax:
   prxmatch(perl-regular-expression, source)
   where:
   i. perl-regular-expression – specifies a character constant, variable, or expression with a value that is a Perl regular expression with syntax:
   /<pattern>/
   ii. source – specifies a character constant, variable, or expression that you want to search.
   iii. If the pattern is found, then prxmatch() returns the position.
   iv. If the pattern is not found, then prxmatch() returns a zero.

The regex_learning_tool project uses a prxmatch() function to match input_file's regex field to its source field.
Syntactically, it would look like this:
prxmatch(regex, source)
The regex_learning_tool project takes the prxmatch() output and displays it with more detail in the following 2 output fields:

1. match_results
   i. If a match exists, then this field will contain the following matched segment info (i.e., info about the source segment that matches the pattern):
   yes, <segment_begin_position>-<segment_end_position>-<segment_length>
   ii. If no match exists, then this field will contain the word 'no.'
2. matched_segment
   i. This is the entire matched segment from beginning to end.
   ii. If no match exists, then this field will be empty.
   iii. Obviously, trailing space(s) will not be seen in this field, but the match_results field will indicate if trailing space(s) exist.

HOW THE PROJECT SIMULATES SAS'S PRXCHANGE() FUNCTION

There are 2 types of match records contained in input_file.xlsx:

1. PRX matches
2. Substitution-PRX matches.

If a match record in input_file.xlsx contains a regex field with the following syntax:
s/<pattern>/<replacement_text>/
then

1. It is a substitution-PRX match.
2. The regex_learning_tool project will use a prxchange() function to process it.
3. The following 2 output fields will contain data:
   i. single_match_replacing
   ii. multiple_match_replacing
   NOTE: These 2 fields will only contain data for a substitution-PRX match.

The following is from SAS Help and Documentation:
SAS's prxchange() function:

1. Performs a pattern-matching replacement.
2. Has the following syntax:
   prxchange(substitution-perl-regular-expression, times, source)
   where:
   i. substitution-perl-regular-expression – specifies a character constant, variable, or expression with a value that is a substitution Perl regular expression with syntax:
   s/<pattern>/<replacement_text>/
   ii. times – is a numeric constant, variable, or expression that specifies the number of times to search for a match and replace a matching pattern.
   If this value is '-1', then matching patterns continue to be replaced until the end of source is reached.
   NOTE: The prxchange() function contained within the regex_learning_tool allows only 2 values for this field: 1 and -1.
   iii. source – specifies a character constant, variable, or expression that you want to search.

iv. If the pattern is found, then prxchange() returns the value in source with the changes that were specified by the substitution perl regular expression.
v. If the pattern is not found, then prxchange() returns the unchanged value in source.

NOTE: Match records with identical patterns occurring within a perl regular expression and substitution perl regular expression will have identical output in their match_results and matched_segment fields.

The regex_learning_tool project uses 2 prxchange() functions to match input_file's regex field to its source field with their results ending up in the following two output fields:

1. single_match_replacing
2. multiple_match_replacing

Syntactically, the prxchange() functions would look like this:

1. prxchange(regex, 1, source)
   The results of this function will be contained in the single_match_replacing output field.
2. prxchange(regex, -1, source)
   The results of this function will be contained in the multiple_match_replacing output field.

THE INPUT: EXCEL FILE INPUT_FILE.XLSX

This is the Excel file (pictured below) that contains all your PRX and substitution-PRX match records and is imported by the project.

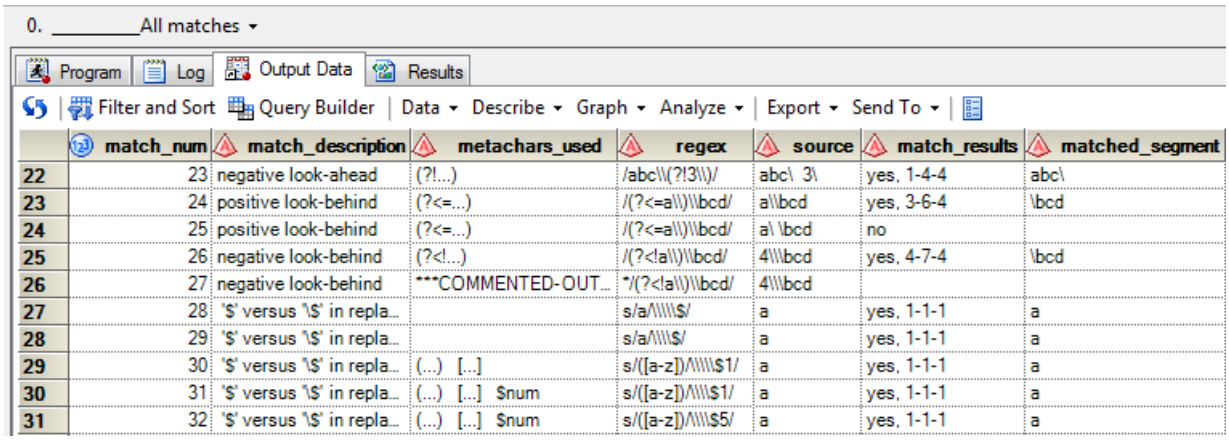
	A	B	C	D
1	match_description	regex	source	notes
23	negative look-ahead	/abc\\(?!3\\)/	abc\ 3\	Matches because 'abc\' is not IMMEDIATELY followed by '3\'
24	positive look-behind	/(?<=a\\)\\bcd/	a\\bcd	Matches because '\bcd' is immediately preceded by 'a\'
25	positive look-behind	/(?<=a\\)\\bcd/	a\ \bcd	No match because '\bcd' is not IMMEDIATELY preceded by 'a\'
26	negative look-behind	/(?<!a\\)\\bcd/	4\\\bcd	Matches because '\bcd' is not immediately preceded by 'a\'
27	negative look-behind	*/(?<!a\\)\\bcd/	4\\\bcd	Matches because '\bcd' is not immediately preceded by 'a\'
28	'$' versus '\$' in replacement text	s/a/\\\\$/	a	'$' equals '\$' in replacement text (ie a '$' in replacement text
29	'$' versus '\$' in replacement text	s/a/\\\$/	a	'$' equals '\$' in replacement text (ie a '$' in replacement text
30	'$' versus '\$' in replacement text	s/([a-z])/\\\\$1/	a	The '$' is backslashed so this is a literal '$'
31	'$' versus '\$' in replacement text	s/([a-z])/\\\$1/	a	The '$' is not backslashed so '$1' means capture buffer 1.
32	'$' versus '\$' in replacement text, ca	s/([a-z])/\\\$5/	a	

This file contains the following fields:

1. match_description
   a. Used for a brief description of the match records.
   b. Entering data for this field is optional.
2. regex
   a. Contains the perl regular expression as it would appear in the 1st argument of a prxmatch() function within a SAS program.
   b. The project allows the commenting-out of this field with the addition of an asterisk (i.e. *) as the first non-whitespace character.
   c. Although prepending an 'm' to your perl regular expression is syntactically correct and can be used in a prxmatch() function within SAS, it is not allowed by the regex_learning_tool. So although 'm/<regex>/' is the same as /<regex>/, use the /<regex>/ syntax.
3. source
   a. Contains the source string as it would appear in the 2nd argument of a prxmatch() function within a SAS program.
   b. As with any character string field in SAS, this field gets padded with trailing spaces.
4. notes
   a. Used for a more detailed description of the how and why for match record results.
   b. Entering data for this field is optional.

THE OUTPUT: SAS DATASET AND SAS REPORT

The output SAS dataset and SAS report contain the same data. Below is sample SAS dataset output (the matched_segment field is repeated for easier viewing) containing the match results.



	match_num	match_description	metachars_used	regex	source	match_results	matched_segment
22	23	negative look-ahead	(?!...)	/abc\\(?!3\\)/	abc\ 3\	yes, 1-4-4	abc\
23	24	positive look-behind	(?<=...)	/(?<=a\\)\\bcd/	a\\bcd	yes, 3-6-4	\bcd
24	25	positive look-behind	(?<=...)	/(?<=a\\)\\bcd/	a\ \bcd	no	
25	26	negative look-behind	(?<!...)	/(?<!a\\)\\bcd/	4\\\bcd	yes, 4-7-4	\bcd
26	27	negative look-behind	***COMMENTED-OUT...	*/(?<!a\\)\\bcd/	4\\\bcd		
27	28	'$' versus '\$' in repla...		s/a/\\\\$/	a	yes, 1-1-1	a
28	29	'$' versus '\$' in repla...		s/a/\\\$/	a	yes, 1-1-1	a
29	30	'$' versus '\$' in repla...	(...) [...]	s/([a-z])/\\\\$1/	a	yes, 1-1-1	a
30	31	'$' versus '\$' in repla...	(...) [...] $num	s/([a-z])/\\\$1/	a	yes, 1-1-1	a
31	32	'$' versus '\$' in repla...	(...) [...] $num	s/([a-z])/\\\$5/	a	yes, 1-1-1	a

matched_segment	single_match_replacing	multiple_match_replacing	notes
abc\			Matches becaus...
\bcd			Matches becaus...
			No match becau...
\bcd			Matches becaus...
			Matches becaus...
a	\\$	\\$	'$' equals '\$' in r...
a	\\$	\\$	'$' equals '\$' in r...
a	\\$1	\\$1	The '$' is backsla...
a	\\a	\\a	The '$' is not bac...
a	\\	\\	

The above SAS output contains the following 10 fields:

1. match_num
2. match_description
3. metachars_used
4. regex
5. source
6. match_results
7. matched_segment
8. single_match_replacing
9. multiple_match_replacing
10. notes

Here are descriptions of these 10 fields:

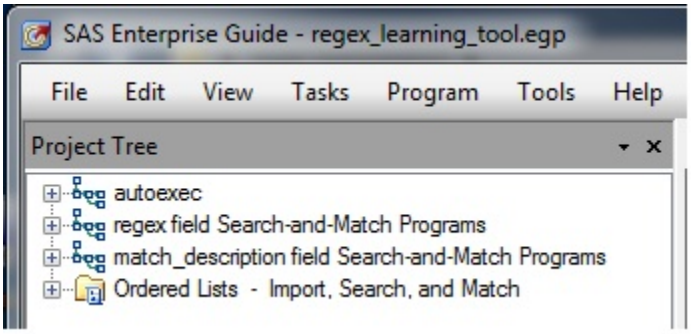
1. match_num
   a. This is input_file's record number for the match record.
   b. This field starts counting at record number 2 since input_file.xlsx has the column headings in record number 1.
2. match_description
   a. Duplicate field from input_file.xlsx
3. metachars_used
   a. This is a list of all metachars being used in the regex field.
   b. For example, an escaped asterisk (i.e. *) occurring in your perl regular expression means a literal asterisk and not the metachar while an escaped backslash followed by an asterisk (i.e. *) means the metachar (i.e. 0 or more backslashes).
   c. This field contains output for any match record whether it was selected as the result of a regex field search or a match_description field search.
4. regex
   a. Duplicate field from input_file.xlsx
   b. Please see the section titled 'THE INPUT: EXCEL FILE INPUT_FILE.XLSX' for special situation involving use of leading 'm' in the perl regular expression.
5. source

a. Duplicate field from input_file.xlsx

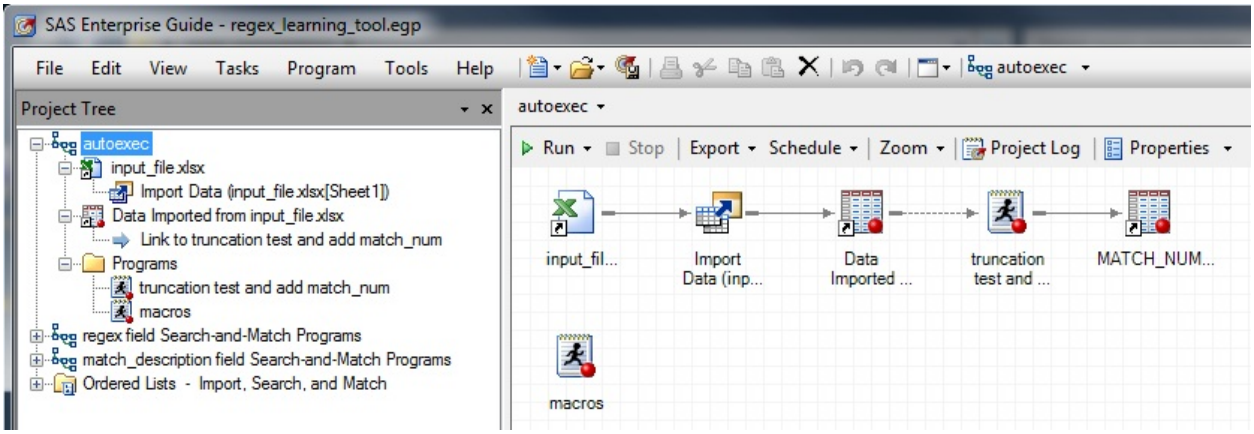
6. match_results
   a. Either 'yes' a match exists or 'no' a match does not exist.
   b. If a match is successful, then the following numbers are listed (separated by hyphens) for the matched segment contained in source:
      1) start location
      2) end location
      3) length of matched segment
7. matched_segment
   a. The actual segment contained in source that matched your perl regular expression.
   b. This field will not show trailing spaces. The match_results field will indicate if trailing spaces exist.
8. single_match_replacing
   a. This field contains output only when the regex field contains a substitution perl regular expression (i.e., the regex field contains syntax: s/<pattern>/<replacement_text>/).
   b. This field contains the results of when only the first occurrence of a match is replaced in source.
   c. Results are the same as using the prxchange() function with a 1 as the 2nd argument like so:
      results = prxchange(substitution_regex, 1, source);
9. multiple_match_replacing
   a. This field contains output only when the regex field contains a substitution perl regular expression (i.e., the regex field contains syntax: s/<pattern>/<replacement_text>/).
   b. This field contains the results of when all occurrences of a match are replaced in source.
   c. Results are the same as using the prxchange() function with a -1 as the 2nd argument like so:
      results = prxchange(substitution_regex, -1, source);
10. notes
    a. Duplicate field from input_file.

THE PROJECT'S 3 PROCESS FLOWS AND 1 ORDERED LIST SECTION

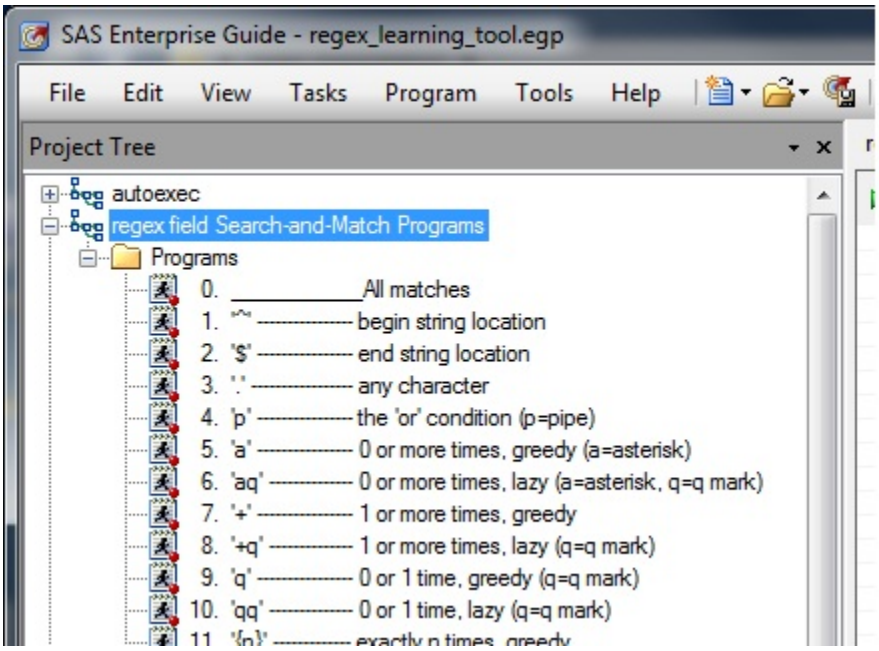
The regex_learning_tool project contains 3 process flows and 1 ordered list section (all pictured below).



Contained within the autoexec process flow window (pictured below) are an import-related task and program as well as a SAS program containing macro programs and variables.



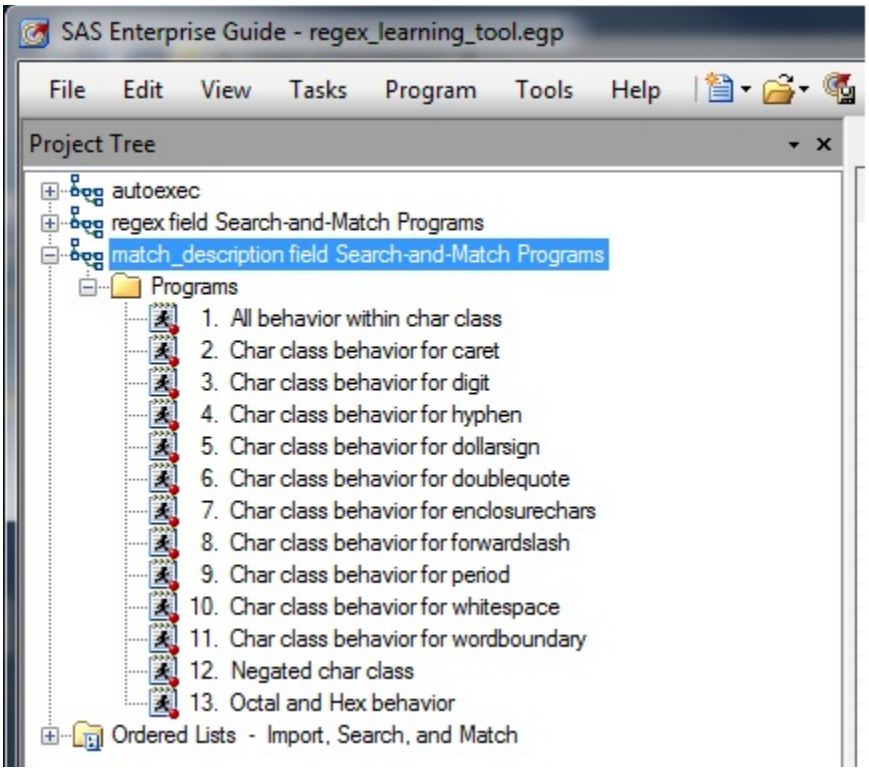
The “regex field Search-and-Match Programs” process flow (pictured below) contains a list of 44 SAS programs that can be used for searching the regex field for PRX metacharacters and then processing these selected match records.



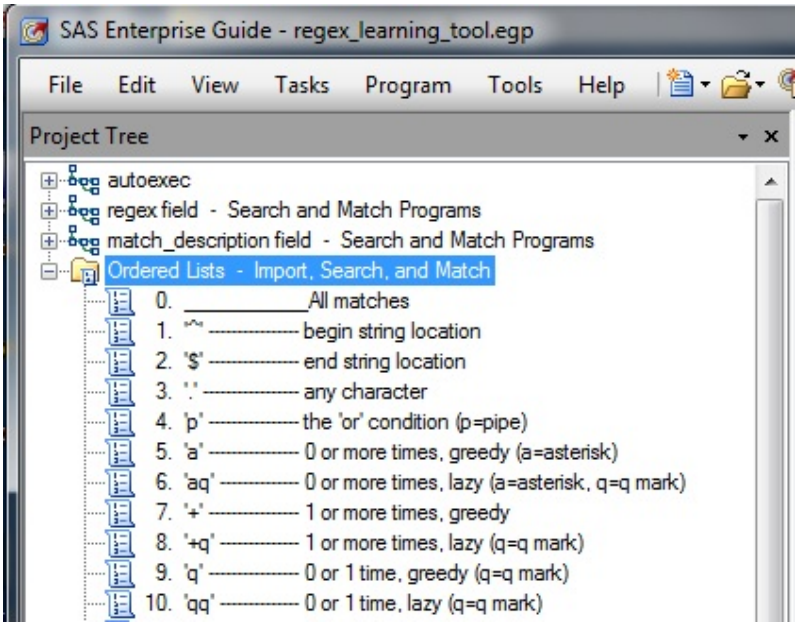
In the above SAS program names, some ascii characters can’t be used for program naming. The following is a list of abbreviations used in place of these characters:

1. p = ‘|’ (i.e., vertical bar or pipe)
2. a = ‘*’
3. q = ‘?’
4. c = ‘:’
5. [b] = ‘\’
6. l = ‘<’ (i.e., l as in less than)

The “match_description field Search-and-Match Programs” process flow (pictured below) contains a list of 13 SAS programs that can be used for searching the match_description field for keywords and then processing these selected match records.



The ordered list section named "Ordered Lists – Import, Search, and Match" (pictured below) contains ordered lists of which each has a corresponding "same-named" SAS program in one of the above 2 process flows.



THE AUTOEXEC PROCESS FLOW

The autoexec process flow contains the following:

1. An Import Data task, which imports the Excel file input_file.
2. The SAS program "truncation test and add match_num," which does the following:
   a. Tests for possible field truncation resulting from the import process.

b. Adds a new field named match_num, which is the record number of the match record within the Excel file input_file. This enables easier referencing of the match record when looking back and forth between input_file.xlsx and either the output SAS dataset or SAS report.

3. The SAS program named "macros," which contains macro variables and macro programs.

THE ORDERED LIST SECTION

The ordered list section:

1. Is automatically created by SAS Enterprise Guide when a project's first ordered list is created.
2. Is used for storing all ordered lists within the project.
   a. The ordered list section contains 57 ordered lists.
   b. The first 44 ordered lists involve regex field searches.
      - A regex field search involves selecting match records based on a search of their regex field for certain PRX metacharacters.
   c. The last 13 ordered lists involve match_description field searches.
      - A match_description field search involves selecting match records based on a search of their match_description field for certain keywords.

THE RELATIONSHIP BETWEEN ORDERED LISTS AND SAS PROGRAMS

As seen in the above pictures, each ordered list has a corresponding "same-named" SAS program located in one of two process flows. Each ordered list, when run, does the following in the order listed:

1. Runs the Import Data task and imports the data contained in the Excel file input_file.xlsx
2. Runs the SAS program named "truncation test and add match_num."
3. Runs the corresponding "same-named" SAS program.
4. Although it is possible to run the "same-named" SAS program directly, an import of input_file.xlsx will not occur and you will not process the latest changes to input_file.xlsx, if any changes existed.
   NOTE: An ordered list should only be run when an import of input_file.xlsx is needed. If an import is not needed, then just run the SAS program by itself and save processing time by skipping the Import Data task.

HOW THE PROJECT PROCESSES A SUBSTITUTION PRX MATCH RECORD

A substitution perl regular expression differs from a perl regular expression in the following ways:

1. Operation
   a. Perl regular expression – allows for only matching of the source field.
   b. Substitution perl regular expression – allows for both the matching and modification of the source field.
2. SAS functions used
   a. Perl regular expression – uses the prxmatch() function with the following syntax:
      prxmatch('/<regex>/', source)
   b. Substitution perl regular expression – uses the prxchange() function with the following syntax:
      prxchange('s/<sub_regex>/<replacement_text>/', times, source)
      where times = the number of times to search for a match and replace a matching pattern

For the regex_learning_tool project, the 2nd argument to the above prxchange() function is restricted to 2 values:

1. 1 = the matching pattern is replaced 1 time and the resulting character string is contained in the single_match_replacing field of the output SAS dataset and SAS report.
2. -1 = the matching patterns continue to be replaced until the end of source is reached and the resulting character string is contained in the multiple_match_replacing field of the output SAS dataset and SAS report.

When the regex field of a match record contains a substitution perl regular expression, then the following 2 fields (of the output SAS dataset and SAS report) will contain output:

1. single_match_replacing
2. multiple_match_replacing

A REGEX FIELD SEARCH VERSUS A MATCH_DESCRIPTION FIELD SEARCH

Both a regex field search and a match_description field search save processing time by only processing selected match records.

1. A regex field search:
   a. Involves selecting match records based on a search of their regex field for certain PRX metacharacter(s).
   b. Is ideal for use when the user can only recall a certain metacharacter that he or she remembers being used in the regex field of the desired match record.
2. A match_description field search:
   a. Selects match records based on a search of their match_description field for certain keyword(s).

b. Is ideally suited for use in which the user is testing a series of related regular expressions that together can be described as a kind of "learning trail."

3. Because of the complexity involved, it is not recommended that the user create any new regex field searches. The regex_learning_tool project has 44 regex field searches for most all PRX metacharacters.
4. To run either a regex field search or a match_description field search, see the section above titled "RUNNING THE PROJECT" and pick up with step #4.

There are some things that need to be stated concerning a match_description field search:

1. It's case-insensitive.
2. There are 3 types of match_description field searches:
   i. "single" keyword search – involves a search of the match_description field for the existence of a single keyword.
   ii. "and" keyword search – involves a search of the match_description field for the existence of 2 or more keywords in which all of the keywords must exist.
   iii. "or" keyword search – involves a search of the match_description field for the existence of 2 or more keywords in which any of the keywords can exist.

   NOTE: "and" searches can't be combined with "or" searches
3. For instructions on how to create a new match_description field search, see the section below titled "CREATING A NEW MATCH_DESCRIPTION FIELD SEARCH."

WHY THE PROJECT MAKES 2 PASSES THROUGH THE DATA

The regex_learning_tool makes the following 2 passes through the imported input_file data containing your match records:

1. The first pass selects all match records fulfilling the search criteria (i.e., either a regex field search or a match_description field search).
2. The second pass creates a list of all prx metacharacters occurring in the perl regular expression (i.e., the regex field) of the selected match records. This list is located in the metachars_used field of the output.

COMMENTING OUT A MATCH RECORD TO AVOID PROCESSING

Instructions for commenting out a match record:

1. Commenting out a match record involves adding a '*' as the first visible (i.e., non-whitespace) character in the regex field. Here are some examples:
   a. */aaa/
   b. *s/a/b
   c. * /aaa/
2. Commenting out a match record:
   a. Prevents the regex_learning_tool project from processing this match record since processing it could cause the project to terminate.
   b. Allows the user to save the uncompleted regex field contents for completion at a later time.
3. A commented-out match record will still be eligible for selection when doing a regex field search or a match_description field search. However, the match record will not be processed and the metachars_used field will contain only:
   COMMENTED-OUT

USING THE MODIFY-SAVE-RUN-REPEAT TECHNIQUE FOR QUICK LEARNING

Instructions for using this technique:

1. Open the Excel file input_file.xlsx, and add a new match record by either:
   a. Copying an existing match record and modifying it.
   b. Inserting a new one.
2. Add an identifying string to the match_description field of the match records that are related in some way.
   a. For example, if you are experimenting with using the grouping metacharacters (i.e., "(" and ")" used in (pattern)) for use in modifying phone numbers, then add the following string to all match records dealing with this experimenting:
      grouping_phone_numbers
   b. Doing this will allow you to quickly process only the match records in input_file.xlsx that have this string in their match_description field.
3. Refer to the below instructions on "creating a new match_description field keyword search" and substitute any of the following for the macro variable character_string_list in the call to the macro grab_recs_match_descr_field:
   a. grouping_phone_numbers
      - In this example, the keyword would be the entire string.
   b. grouping@phone_numbers

- In this example, both keywords must exist in the match_description field in order for the match record to be selected for processing.

c. grouping@phone@numbers
- In this example, all 3 keywords must exist in the match_decription field in order for the match record to be selected for processing.

4. When you are finished creating a new match_description field search, then you can repeat the following 3 steps as often as you like in order to quickly see the effects of any change to a match record in input_file.xlsx:
   a. Create a new match record by making a copy of an existing match record and modifying it.
   b. Save the Excel file input_file.xlsx (otherwise, your changes won't be imported).
   c. Run the ordered list for this match_description field search.

CREATING A NEW MATCH_DESCRIPTION FIELD SEARCH

Instructions for creating and running a new match_description field search.

1. A match_description field search consists of the following:
   a. A SAS program.
   b. An ordered list containing 3 programs/tasks the last of which is the SAS program.
2. Creating the SAS program
   a. In the "match_description field Search-and-Match Programs" process flow, open a new SAS program and paste the following code:
      %grab_recs_match_descr_field(<character string list>, <dataset name>)
      %match_regex(<dataset name>)
   b. <character string list> must be either:
      1) A list of character strings separated by one of the following delimiters:
         a) @ – indicates an "and search."
         b) ! – indicates an "or search."
      2) A single character string containing no '@' or '!' delimiters.
   c. <dataset name> must be:
      1) The character string 'DATA_' with an appended number.
      2) Make sure that other match_description field search programs do not have the same number being appended to 'DATA_.' Increment this number each time that a new match_description field search program is added to the "match_description field Search-and-Match Programs" process flow.
   d. Here is an example SAS program:
      %grab_recs_match_descr_field(behavior_within_char_class@caret, data_2)
      %match_regex(data_2)
3. Creating the ordered list
   a. From any process flow, do the following:
      1) Right-click within the process flow window area.
      2) Click on New->Ordered List.
      3) From the Ordered List window, add the 3 programs/tasks in the following order:
         i. Import Data (input_file.xlsx[Sheet 1]) (an Import Data task)
         ii. Truncation test and add match_num (a SAS program)
         iii. The SAS program that you just created.
         iv. Click on the Save button.
   b. You are now ready to run your new match_description field search.
4. To run your new match_description field search, see the above section titled "RUNNING THE PROJECT" and pick up with step #4

WHAT TO DO IF YOU GET A FIELD TRUNCATION ERROR

A field truncation error:

1. Informs the user that at least one match record contained in input_file.xlsx is having one of its 4 fields truncated during the import process.
2. Is indicated in the SAS log in the following manner:
   ERROR: The <insert input_file field> field is being truncated.
   ERROR: Please increase field length and rerun.

To remedy a field truncation error, do the following:

1. From the autoexec process flow window, double-click on the 'Import Data' task.
2. Click on 'Modify Task.'
3. Keep clicking the 'Next>' button until you get to the 'Define Field Attributes' window.
4. Highlight the field that is being truncated and click the 'Modify…' button.

5. In the 'Field Attributes for <insert field name>' window, increase the Length field value as well as the width values for any of the other listed formats and informats.
6. Click on 'Finish' to run the task.

CONCLUSION

Practicing is the key to learning PRX matching within SAS. The ability to test, modify, and retest a perl regular expression and log your entire practice trail is essential to both learning and retaining. This tool allows the user to keep all perl regular expression learning in one Excel file. And don't worry about managing the file if it becomes messy and cluttered. Let the regex_learning_tool project do that for you.

REFERENCES

SAS Help and Documentation: Tables of Perl Regular Expression (PRX) Metacharacters

ACKNOWLEDGMENTS

I would like to thank the following people:
Lt Col Melinda Eaton, USAF for her help with editing this paper.
Mr. James Escobar, MPH for his help with editing this paper.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Paul Genovesi
Henry Jackson Foundation for the Advancement of Military Medicine Inc.
Wright-Patterson AFB, Ohio
E-mail: pcg7285g@gmail.com